Automated Simplification of Large Symbolic Expressions

David H. Bailey

Lawrence Berkeley National Laboratory,
Berkeley, CA 94720

Jonathan M. Borwein

Centre for Computer Assisted Research Mathematics and its Applications (CARMA), Laureate Professor, Univ of Newcastle, Callaghan, NSW 2308, Australia and Distinguished Professor, King Abdul-Aziz Univ, Jeddah.

Alexander D. Kaiser

Courant Institute of Mathematical Sciences, New York University,
New York, NY 10012

Abstract

We present a set of algorithms for automated simplification of constants of the form $\sum_i \alpha_i x_i$ with α_i rational and x_i complex. The included algorithms, called `SimplifySums` and implemented in *Mathematica*, remove redundant terms, attempt to make terms and the full expression real, and remove terms using repeated application of the integer relation detection algorithm PSLQ. Also included are facilities for making substitutions according to user-specified identities. We illustrate this toolset by giving some real-world examples of its usage, including one, for instance, where the tool reduced a symbolic expression of approximately 100,000 characters in size enough to enable manual manipulation to one with just four simple terms.

Key words: Simplification, Computer Algebra Systems, Experimental Mathematics, Error Correction

---

⋆ Supported in part by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy, under contract number DE-AC02-05CH11231. (David H. Bailey)

Email addresses: `dhbailey@lbl.gov` (David H. Bailey), `jonathan.borwein@newcastle.edu.au` (Jonathan M. Borwein), `adkaiser@gmail.com` (Alexander D. Kaiser).

URLs: `http://crd.lbl.gov/~dhbailey/` (David H. Bailey), `http://carma.newcastle.edu.au/jon/`

1. Introduction

A common occurrence for many researchers who engage in computational mathematics is that the result of a *Mathematica* or *Maple* operation is a very long, complicated expression, which although technically correct, is not very helpful; only later do these researchers discover, often indirectly, that in fact the complicated expression they produced further simplifies, sometimes dramatically, to something much more elegant and useful. Such instances are to be expected, given the limitations of any symbolic computing package, for any of a number of reasons, including the difficulty of recognizing when a given subexpression is zero.

Such instances are closely related to the problem of recognizing a numerical value as a closed-form expression. In this case, researchers have used integer relation-finding algorithms such as the PSLQ algorithm (Bailey and Broadhurst, 2000) to express the given numerical value as a linear sum of constants or terms. In both instances, researchers seek as simple a closed-form expression as possible. Such simplified closed-form expressions are highly desirable, both in mathematical research and in problems, say, from mathematical physics.

The importance of closed forms is described in (Borwein and Crandall, Accepted May 2010), and examples of such work are described in (Bailey et al., 2010b) and (Borwein et al., 2010).

We present herein a software package `SimplifySums` for the simplification of constants of the form $\sum_i \alpha_i x_i$, where each α_i is rational and each x_i real or complex. Such constants frequently arise in looking for closed forms for integrals or sums, and are frequently large and machine-generated by symbolic mathematics software such as *Mathematica* or *Maple*.

Implemented in *Mathematica*, the package includes a focused set of tools for simplification of such constants. The package is able to remove redundant constants, opposites and conjugates, symbolically, numerically or both. It can simplify complex terms, which is useful if some x_i are complex yet the whole constant is real. The package uses symbolic algebra to apply the integer relation detection algorithm PSLQ repeatedly and utilize results expression with exact, rational number arithmetic. It also contains code to apply substitutions, so the user may specify identities or substitutions they would like performed. The tools can be accessed through a convenient, simple function interface. Users also have access to the individual functions that can be customized.

The criterion to decide which version of an expression is simpler is straightforward — a sum that has fewer terms is simpler. [1] All the algorithms (with the exception of substitution) only process the rational coefficients α_i, so an expression $\sum_{i=1}^m \alpha_i x_i$ is simpler than $\sum_{i=1}^n \beta_i x_i$ if $m < n$. This is in contrast to the general case discussed in (Carette, 2004), where the question of which version of a general expression is considered and formalized.

The tools have proven quite effective. Many computer-generated constants have many of the simple redundancies described above. The techniques using integer-relations are general and reliable, provided numerical results are used with appropriate caution. The substitutions allow the user to apply specific identities automatically. This will allow them to use identities that arise repeatedly in particular work, but are not in *Mathematica*.

(Jonathan M. Borwein), `http://hpcrd.lbl.gov/~adkaiser/` (Alexander D. Kaiser).

[1] For short expressions this may occasionally lead to less elegant presentation; for longer ones it seems highly desirable.

Note also that the restriction to sums is very general — no limit is placed on each x_i, only that it must evaluate to a real or complex number, so each x_i may be arbitrarily complicated. Each term will be treated as a single constant by all parts of the code, with the exception of substitutions.

The remainder of the paper is structured as follows. In Section 2 existing literature on simplification and simplification in current CAS is discussed. In Section 3 the general structure of `SimplifySums` is described. Precise descriptions of the package are relegated to an Appendix (Section 6). In Section 4 we give a variety of examples and conclude in Section 5 with some discussion of future research directions.

Note that all tests were performed on a stock MacBook Pro with a 2.53 GHz Intel Core 2 Duo processor and 4 GB of RAM using *Mathematica 7.01*.

2. Related Literature and Previous Work

There are two central questions to consider when designing a simplification algorithm. First, what does it mean for an expression to be simpler than another? Second, given a constant, what algorithms can be applied to make the expression simpler?

2.1. Simplification in the literature

The paper (Carette, 2004) provides a formal description of simplification. The author discusses, using ideas including Kologorov complexity and minimum description length, a method for defining whether a version of an expression is more simple than another. The author also discusses use of this formalism to make practical decisions on simplification of particular expressions and discusses the relationship of his formalism and the simplification algorithms included with *Maple*. Notably, the author also discusses the lack of available literature, both on formalism and practical methods for simplificaion: "But if one instead scours the scientific literature to find papers relating to simplification, a few are easily found: a few early general papers... some on elementary functions... as well as papers on nested radicals... Looking at the standard textbooks on Computer Algebra Systems (CAS) leaves one even more perplexed: it is not even possible to find a proper definition of the problem of simplification."

Searching for methods of simplification reveals many older papers as mentioned in (Carette, 2004). The papers (Buchberger and Loos, 1982; Casas et al., 1990; Caviness, 1970; Fateman, 1972; Fitch, 1973; Moses, 1971) explore formalism and technique for simplification. The papers (Caviness and Fateman, 1976; Zippel, 1985) discuss simplification techniques specific to expressions involving radicals. The papers (Harrington, 1979; Hearn, 1971) discuss an earlier CAS called *Reduce* and some associated algorithms. All of these provide relevant early discussions of the basic questions here, but there have been dramatic advances in CAS systems and computing power since they were written.

For more modern techniques, there is a variety of literature on theoretical matters of simplification, and much on simplification and resolution of specific types of expressions. The work (Stoutemyer, 2011) describes the philosophy and goals of a practical, effective simplification algorithm, discussing many heuristics about correctly selecting branches, merits of particular forms of various expressions and user control and interface. The papers (Bradford and Davenport, 2002; Beaumont et al., 2003, 2004) primarily address simplification of elementary functions in the presence of branch cuts, building on the earlier work (Dingle and Fateman, 1994), though none of these address practical issues associated with large expressions. The work (Schneider, 2008) deals with a specific class of symbolic sums, in particular the question of finding closed forms of sums dependent on a parameter, and (Kauers, 2006) approaches the same problem for a more general class of symbolic sums. The work (Gutierrez and Recio, 1998) describes simplification of highly specific expressions involving sines and cosines related to inverse kinematic problems. The work (Monagan and Pearce, 2006) discusses simplification specific to rational expressions modulo an ideal of polynomial rings. The work (Fateman, 2003) discusses how to check automatically that a program is correct, and explores certain questions of automatic simplification that occur in the process.

2.2. Simplification in current CAS

Two of the most commonly used simplification routines are *Mathematica*'s Simplify and FullSimplify. The system is closed and proprietary; documentation of the algorithms is not available. Empirically, *Mathematica*'s Simplify and FullSimplify tend to get "gummed up" when run on a very large sum and become so slow they sometimes do not return results for over a day or ever. Neither algorithm returns any intermediate updates, leading one to wonder after over a day if anything will ever return. It is not clear why this is true or if there are effective, general ways to combat these problems. Regardless, these routines were inadequate to simplify constants that arose in work such as (Bailey et al., 2010b; Borwein et al., 2010).

In *Maple* (Maple, 2012) more documentation is available but the underlying issues remain. More details are available about customization of the algorithms, one can instruct the algorithm to focus on exponential, logarithmic or rational functions, or specify expressions in polar coordinates. Notably, one may specify that the given expression is a constant not dependent on any parameters and issue a preference for reducing the size of such an expression. In this context, the algorithms will also look for possible cancelations involving the real and imaginary parts of complex subexpressions. The algorithm also leverages numerical information, but the precise method of this is not stated. Some information is given in the documentation on nesting strategies, but again, not enough to truly understand internals of the algorithms.

SAGE (Stein et al., 2012), which is free and open source, relies on another CAS called *Maxima* (Maxima, 2011) for simplification algorithms. Documentation for *Maxima* describes routines for symbolic summation, simplification of rational functions, and various facilities for user defined patterns. The SAGE and *Maxima* documentation do not state high level simplification strategies, and the source code is difficult to follow.

In general, it seems there is very little modern literature on how to build or implement a simplification routine for the case of a large, machine generated input constant. When one has a sum of the form considered here, robustness in the presence of hundreds or thousands of terms is crucial. Moderate scaling of runtime is not a problem, but scaling of runtime that leads the user to think nothing is happening for hours on end is unacceptable. The `SimplifySums` toolset is designed to address these concerns. By focusing on sums, we can employ such straightforward and effective algorithms.

In summary, we believe that the current package occupies a useful and currently unfilled space among existing simplification packages and algorithms.

3. The '`SimplifySums`' package

The package components are as follows, all steps of which can be called separately.

(1) First, *redundancy* is explored. The code compares all pairs to remove redundant equalities, opposites and complex conjugates.
  - This $O(n^2)$ loop is robust at removing such elements, while more generic approaches may miss such relationships or simply fail to function. The loop repeats until no change is detected.

(2) Next are *decomplexification* routines to attempt to make constants real.
  - The code looks for terms that are stored as complex but have zero imaginary part and convert them to real datatypes. It will then evaluate remaining complex terms and convert them to real if their imaginary parts sum to zero, or remove them from the sum entirely if both the real and imaginary parts sum to zero.

(3) The code will then run an *integer-relation detection* step using the algorithm PSLQ (Bailey and Broadhurst, 2000) to remove dependent terms.
  - An *integer relation algorithm* takes a list of real or complex numbers $(x_1, x_2 \dots x_n)$ and attempts to find a nontrivial relationship

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n = 0 \tag{1}$$

    where each α_i is an integer. See (Bailey and Broadhurst, 2000) for details on the algorithm. If such a relationship is found, the code will use the identity

$$x_i = -\frac{\sum_{j, j \neq i} \alpha_j x_j}{\alpha_i} \tag{2}$$

    for some i such that $\alpha_i \neq 0$ to remove x_i from the expression.
  - By default, 500-digit arithmetic is used in the PSLQ routine, and it is presumed that an identity that holds to 500-digit arithmetic is in fact a true mathematical identity, even though in a strict mathematical sense this cannot be guaranteed. If a higher level of certitude is desired, the precision level can be increased. Relationships with coefficients with too large a norm are thrown out. However, *Mathematica* handles rational coefficients with exact arithmetic. Thus, if the relationship is valid, there are no numerical errors made performing this substitution.
  - Since the runtime scaling of this algorithm can be dramatic, by default the code splits the sum into blocks of 10, 20 then 50 adjacent elements of the sum.

(4) As an alternative to step 3, the code will select a subset of elements according to user specified categories and apply PSLQ. It will then perform a substitution according to the relationship found, then repeat this process until no relationships are found.
  - Empirically, relationships are much more likely to be found between terms that have some mathematical relationship with each other. Thus, the code can examine subsets that are related according to user provided categories. For example, as in Example 6 below one may wish to group all terms with logarithms in one category, dilogarithms in another and so forth.
  - As in step 3, within each category PSLQ is run in succession on 10, 20 then 50 adjacent elements of the sum.

(5) *Zero determination* is treated separately. Alternately, the code will also check for the fortunate circumstance that

$$\sum_{\substack{i \text{ s.t.} \\ \alpha_i \neq 0}} x_i = 0. \tag{3}$$

- That is, some combination of terms in the original equation simply summed to zero. In this case the appropriate x_i are all removed. This is surprisingly common in practice with machine generated constants. The use of *PSLQ* reveals such relationships.
- As mentioned in the previous item, conservative parameters choices are used to ensure that the relationships discovered are valid, as best as can be checked numerically.

(6) Also included is a routine that will make a *randomized selection* of terms, then run PSLQ repeatedly as above.

- This is not run by default, but still included in the source. This may be effective when little is known about the individual terms of the sum. If the user has enough knowledge to determine categories *a priori*, then using that knowledge is usually more effective.

(7) The final portion of the code is a *substitution* package.

- This takes a *Mathematica* rule, performs pattern matching and substitutes the user's desired expression. The *Mathematica* documentation pages have far more information on the use of such substitutions. This operates on individual terms of the sum to maintain robustness on very large sums. This is in contrast to built-in routines, which the *Mathematica* documentation says act on "every subpart of your expression." Perhaps due to exponential growth in the number of terms in "every subpart" of a sum, this routine may run slowly.

Remark 1 (Disclaimer)**.** This combination of procedures can be very effective at removing and simplifying terms. The user must, however, be mindful to consider the difference between numerical matches and true equality. Depending on the options used, numerical comparisons may be used repeatedly as 'truths' in this package. Such output must, of course not be taken as proof, only as experimental evidence. But in many applications this may not matter, and in some others "knowledge is nine-tenths of a proof". ◇

4. Examples and results

We now provide a few examples of the type of manipulations that the package can usefully perform.

Example 1 (Logarithms)**.** The expression below contains six complex logarithms, some of which are conjugates and some of which are linearly dependent. The constant is presented as

$$\begin{aligned} C_1 := &-\frac{1}{8} i\pi \log^2\left(\frac{2}{3} - \frac{2i}{3}\right) + \frac{1}{8} i\pi \log^2\left(\frac{2}{3} + \frac{2i}{3}\right) + \frac{1}{12}\pi^2 \log(-1-i) \\ &+ \frac{1}{12}\pi^2 \log(-1+i) + \frac{1}{12}\pi^2 \log\left(\frac{1}{3} - \frac{i}{3}\right) + \frac{1}{12}\pi^2 \log\left(\frac{1}{3} + \frac{i}{3}\right). \end{aligned} \tag{4}$$

This is automatically reduced to

$$\frac{2}{3}\ \mathrm{Re}\left(-\frac{1}{8} i\pi \log^2\left(\frac{2}{3}-\frac{2i}{3}\right)\right)+\ \mathrm{Re}\left(\frac{1}{12}\pi^2 \log\left(\frac{1}{3}-\frac{i}{3}\right)\right), \tag{5}$$

where three terms have been removed by the simple comparisons, and one more through use of PSLQ. This in turn can be manually simplified to

$$-\frac{1}{48}\pi^2 \log(18) \tag{6}$$

on taking the principal branch of log on both terms.

Of course, correctly selecting branches require care, so the code does not perform this particular simplification unless an appropriate rule is set or calls to the *Mathematica* simplify functions are made. ◇

Example 2 (Arctangents)**.** To illustrate the problem consider the arctangent identity

$$\frac{\pi}{2} - \arctan\left(\sqrt{5}\right) = \arctan\left(\frac{3-\sqrt{5}}{4}\right) + \arctan\left(\sqrt{5}-2\right)$$

which when expressed in terms of logarithms is

$$\begin{aligned}
&\frac{1}{2}\, i\left(\log\left(1-\frac{1}{5}\, i\sqrt{5}\right)-\log\left(1+\frac{1}{5}\, i\sqrt{5}\right)\right) \\
&=\frac{1}{2}\, i\left(\log\left(1-\frac{3}{4}\, i+\frac{1}{4}\, i\sqrt{5}\right)-\log\left(1+\frac{3}{4}\, i-\frac{1}{4}\, i\sqrt{5}\right)\right) \\
&+\frac{1}{2}\, i\left(\log\left(1+2\, i-i\sqrt{5}\right)-\log\left(1-2\, i+i\sqrt{5}\right)\right)+\frac{1}{4}\, i\log\left(16+\left(\sqrt{5}-3\right)^2\right) \\
&-\frac{1}{4}\, i\log\left(16+\left(3-\sqrt{5}\right)^2\right)+\frac{1}{4}\, i\log\left(1+\left(2-\sqrt{5}\right)^2\right)-\frac{1}{4}\, i\log\left(1+\left(-2+\sqrt{5}\right)^2\right).
\end{aligned} \tag{7}$$

If presented in just this form, the user or system might well find the simplifications, but if through intervening steps the logarithms have been rearranged and manipulated, or additional terms are added, all bets are off. Even if the expression is found, runtime may increase drastically depending on the algorithms used. Here, `SimplifySums` reduces the right hand side to

$$2\ \mathrm{Re}\left(-\frac{1}{2} i\log\left(\left(1+\frac{3i}{4}\right)-\frac{i\sqrt{5}}{4}\right)\right)+2\ \mathrm{Re}\left(\frac{1}{2} i\log\left((1+2i)-i\sqrt{5}\right)\right)$$

which can be further reduced to $\operatorname{arccot}(\sqrt{5})$, which is equal to the original expression, with FullSimplify. FullSimplify returns the equivalent $1/4\left(\pi-\arctan\left(4\sqrt{5}\right)\right)$.

Consider now expression (8) below. It includes all the terms in (7), adding additional log terms for a total of 23 elements, and has randomly permuting the elements.

$$
\begin{aligned}
&\mathrm{Re}\left(-\frac{1}{4}\log^2\left(-\frac{1}{3}+i\right)\log\left(\frac{1}{3}-i\right)\right)+2\mathrm{Re}\left(\frac{1}{4}\log\left(-\frac{1}{3}+i\right)\log^2\left(\frac{1}{3}-i\right)\right) \\
&+\mathrm{Re}\left(-\frac{1}{8}i\pi\log^2\left(\frac{2}{3}-\frac{2i}{3}\right)\right)-\mathrm{Re}\left(-\frac{1}{8}i\pi\log^2\left(1-\frac{i}{3}\right)\right)+2\mathrm{Re}\left(-\frac{1}{8}i\pi\log^2(1-3i)\right) \\
&+\mathrm{Re}\left(-\frac{1}{4}\log\left(\frac{1}{2}-\frac{i}{2}\right)\log^2(2)\right)-2\mathrm{Re}\left(\frac{1}{2}\log(1-2i)\log^2(2)\right)-4\mathrm{Re}\left(-\frac{1}{4}\log(1-3i)\log^2(2)\right) \\
&+\mathrm{Re}\left(\frac{1}{12}\pi^2\log(-1-i)\right)+\mathrm{Re}\left(\frac{1}{12}\pi^2\log\left(\frac{1}{3}-\frac{i}{3}\right)\right)+2\mathrm{Re}\left(-\frac{1}{2}\log\left(\frac{1}{3}+\frac{i}{3}\right)\log\left(\frac{1}{3}-i\right)\log\left(\frac{2}{3}-\frac{i}{3}\right)\right) \\
&+2\mathrm{Re}\left(\frac{1}{2}\log\left(\frac{1}{3}-\frac{i}{3}\right)\log\left(\frac{2}{3}+\frac{i}{3}\right)\log\left(1-\frac{i}{3}\right)\right)+2\mathrm{Re}\left(\frac{1}{4}i\pi\log(1-3i)\log(2-i)\right) \\
&+\mathrm{Re}\left(\frac{1}{4}\log(1-i)\log(4)\log\left(-\frac{1-\frac{1}{\sqrt{2}}}{-1-\frac{1}{\sqrt{2}}}\right)\right)+\log(2)\log(4)\log\left(-\frac{1-\frac{1}{\sqrt{2}}}{-1-\frac{1}{\sqrt{2}}}\right) \\
&-\frac{1}{2}i\log\left(\left(1+\frac{3i}{4}\right)-\frac{i\sqrt{5}}{4}\right)+\frac{1}{2}i\log\left(\left(1-\frac{3i}{4}\right)+\frac{i\sqrt{5}}{4}\right)+\frac{1}{2}i\log\left((1+2i)-i\sqrt{5}\right) \\
&-\frac{1}{2}i\log\left((1-2i)+i\sqrt{5}\right)+\frac{1}{4}i\log\left(1+\left(2-\sqrt{5}\right)^2\right)-\frac{1}{4}i\log\left(16+\left(3-\sqrt{5}\right)^2\right) \\
&+\frac{1}{4}i\log\left(16+\left(\sqrt{5}-3\right)^2\right)-\frac{1}{4}i\log\left(1+\left(\sqrt{5}-2\right)^2\right)
\end{aligned} \tag{8}
$$

FullSimplify successfully finds a relationship among the logarithms involving arctangents (though not precisely the form above) and reduces the expression to the 14 terms in (9). However, it took 2259.4 seconds, or approximately 38 minutes to produce the following result:

$$
\begin{aligned}
&-\mathrm{Re}\left(\frac{1}{4}\log^2\left(-\frac{1}{3}+i\right)\log\left(\frac{1}{3}-i\right)\right)+2\mathrm{Re}\left(\frac{1}{4}\log\left(-\frac{1}{3}+i\right)\log^2\left(\frac{1}{3}-i\right)\right) \\
&-\mathrm{Re}\left(\frac{1}{8}i\pi\log^2\left(\frac{2}{3}-\frac{2i}{3}\right)\right)+\mathrm{Re}\left(\frac{1}{8}i\pi\log^2\left(1-\frac{i}{3}\right)\right) \\
&-2\mathrm{Re}\left(\frac{1}{2}\log\left(\frac{1}{3}+\frac{i}{3}\right)\log\left(\frac{1}{3}-i\right)\log\left(\frac{2}{3}-\frac{i}{3}\right)\right) \\
&+2\mathrm{Re}\left(\frac{1}{2}\log\left(\frac{1}{3}-\frac{i}{3}\right)\log\left(\frac{2}{3}+\frac{i}{3}\right)\log\left(1-\frac{i}{3}\right)\right)-2\mathrm{Re}\left(\frac{1}{8}i\pi\log\left(\frac{3}{5}-\frac{i}{5}\right)\log(1-3i)\right) \\
&+\frac{\pi}{4}+\frac{5\log^3(2)}{8}+\frac{1}{12}\log(8)\log(512)\log\left(3-2\sqrt{2}\right)-\arctan\left(2-\sqrt{5}\right) \\
&-\frac{1}{2}\arctan\left(\frac{1}{4}\left(\sqrt{5}-3\right)\right)-\frac{1}{2}\arctan\left(3+\sqrt{5}\right)-\frac{1}{6}\pi^2\mathrm{arccoth}(5).
\end{aligned} \tag{9}
$$

By comparison, running `SimplifySums` reduced the sum in (8) to 14 terms (all logarithms in this case) in 1.5 seconds, much faster than 38 minutes for FullSimplify. A further application of FullSimplify requires 61.3 seconds and reduces the outpuut to the following 8 terms.

$$\mathrm{Re}\left(-\frac{1}{4}\log^2\left(-\frac{1}{3}+i\right)\log\left(\frac{1}{3}-i\right)\right)+2\mathrm{Re}\left(\frac{1}{4}\log\left(-\frac{1}{3}+i\right)\log^2\left(\frac{1}{3}-i\right)\right)$$
$$-\frac{1}{3}\mathrm{Re}\left(\frac{1}{8}i\pi\log^2\left(\frac{2}{3}-\frac{2i}{3}\right)\right)+\mathrm{Re}\left(-\frac{1}{8}i\pi\log^2\left(1-\frac{i}{3}\right)\right)+\frac{5\log^3(2)}{8}$$
$$-\frac{1}{48}\pi^2\log\left(\frac{9}{2}\right)+\frac{1}{4}\log(2)\log(512)\log\left(3-2\sqrt{2}\right)+\operatorname{arccot}\left(\sqrt{5}\right). \tag{10}$$

This also illustrates another point — FullSimplify is a powerful routine, and sometimes the best result comes from applying `SimplifySums` and FullSimplify in combination. ◇

Elaborate integrands can arise in high-end use of computer algebra packages. Many of the following examples involve the *polylogarithm* $\mathrm{Li}_n(z) := \sum_{k\geq 1} x^k/k^n$ of order n. (Note that $\mathrm{Li}_1(x) = -\log(1-x)$.)

Example 3 (Integrals I)**.** Consider the following integral, which arose in connection to the integral $\mathcal{K}_1$ in (Bailey et al., 2010a).

$$\int_{\pi/6}^{\pi/3} \log\left(2\sin\left(\frac{x}{2}\right)\right)\,\mathrm{d}x \tag{11}$$

Mathematica evaluates the integral symbolically to

$$\frac{1}{144}\Bigg(-144i\left(\mathrm{Li}_2\left(\sqrt[6]{-1}\right)-\mathrm{Li}_2\left(\sqrt[3]{-1}\right)\right)+19i\pi^2$$
$$+12\pi\left(\log(2)+2\log\left(1-\sqrt[6]{-1}\right)-2\log\left(\sqrt{3}-1\right)\right)\Bigg). \tag{12}$$

Here, *Mathematica* has produced complex subexpressions in evaluating an expression that is real. This is but one simple example of this phenomenon which occurs regularly in computing integrals, including the following examples. After applying `SimplifySums`, we have

$$\mathrm{Re}\left(-i\mathrm{Li}_2\left(\sqrt[6]{-1}\right)\right)+\ \mathrm{Re}\left(i\mathrm{Li}_2\left(\sqrt[3]{-1}\right)\right). \tag{13}$$

In this case, the imaginary parts sum to zero and are removed, removing one term entirely. PSLQ finds that three remaining terms sum to zero and are removed by zero determination. In contrast, FullSimplify reduces the original expression to

$$\frac{1}{16}i\left(16\left(\mathrm{Li}_2\left(\sqrt[3]{-1}\right)-\mathrm{Li}_2\left(\sqrt[6]{-1}\right)\right)+\pi^2\right) \tag{14}$$

which has the disadvantage that it appears complex (though is also real) and has one additional term.

We note that if the user or system is aware of the literature on *logsine* integrals or the *Clausen* function, $\mathrm{Cl}_2(\theta) = \mathrm{Im}\,\mathrm{Li}_2\left(e^{i\theta}\right)$ (Borwein et al., 2012; Lewin, 1981), he or she will immediately reduce (13) to $\mathrm{Cl}_2(\pi/3) - \mathrm{Cl}_2(\pi/6)$. ◇

Example 4 (Integrals II)**.** In work on integrals arising in the Ising model in ferromagmetics, we needed to evaluate an integral E_5, which started life as 4-D integral (Bailey

$$
\begin{aligned}
E_5 = \int_0^1 \int_0^1 \int_0^1 & \big[2(1-x)^2(1-y)^2(1-xy)^2(1-z)^2(1-yz)^2(1-xyz)^2 \\
& \big(- \big[4(x+1)(xy+1)\log(2)\,\big(y^5z^3x^7 - y^4z^2(4(y+1)z+3)x^6 - y^3z\,\big(\big(y^2+1\big)\,z^2+4(y+ \\
& 1)z+5\big)\,x^5 + y^2\,\big(4y(y+1)z^3+3\,\big(y^2+1\big)\,z^2+4(y+1)z-1\big)\,x^4+y\,\big(z\,\big(z^2+4z \\
& +5\big)\,y^2+4\,\big(z^2+1\big)\,y+5z+4\big)\,x^3+\big(\big(-3z^2-4z+1\big)\,y^2-4zy+1\big)\,x^2-(y(5z+4) \\
& +4)x-1)\big]\,/\,\big[(x-1)^3(xy-1)^3(xyz-1)^3\big]+\big[3(y-1)^2y^4(z-1)^2z^2(yz \\
& -1)^2x^6+2y^3z\,\big(3(z-1)^2z^3y^5+z^2\,\big(5z^3+3z^2+3z+5\big)\,y^4+(z-1)^2z \\
& \big(5z^2+16z+5\big)\,y^3+\big(3z^5+3z^4-22z^3-22z^2+3z+3\big)\,y^2+3\,\big(-2z^4+z^3+2 \\
& z^2+z-2\big)\,y+3z^3+5z^2+5z+3\big)\,x^5+y^2\,\big(7(z-1)^2z^4y^6-2z^3\,\big(z^3+15z^2 \\
& +15z+1\big)\,y^5+2z^2\,\big(-21z^4+6z^3+14z^2+6z-21\big)\,y^4-2z\,\big(z^5-6z^4-27z^3 \\
& -27z^2-6z+1\big)\,y^3+\big(7z^6-30z^5+28z^4+54z^3+28z^2-30z+7\big)\,y^2-2\,\big(7z^5 \\
& +15z^4-6z^3-6z^2+15z+7\big)\,y+7z^4-2z^3-42z^2-2z+7\big)\,x^4-2y\,\big(z^3\,\big(z^3 \\
& -9z^2-9z+1\big)\,y^6+z^2\,\big(7z^4-14z^3-18z^2-14z+7\big)\,y^5+z\,\big(7z^5+14z^4+3 \\
& z^3+3z^2+14z+7\big)\,y^4+\big(z^6-14z^5+3z^4+84z^3+3z^2-14z+1\big)\,y^3-3\,\big(3z^5 \\
& +6z^4-z^3-z^2+6z+3\big)\,y^2-\big(9z^4+14z^3-14z^2+14z+9\big)\,y+z^3+7z^2+7z \\
& +1\big)\,x^3+\big(z^2\,\big(11z^4+6z^3-66z^2+6z+11\big)\,y^6+2z\,\big(5z^5+13z^4-2z^3-2z^2 \\
& +13z+5\big)\,y^5+\big(11z^6+26z^5+44z^4-66z^3+44z^2+26z+11\big)\,y^4+\big(6z^5-4 \\
& z^4-66z^3-66z^2-4z+6\big)\,y^3-2\,\big(33z^4+2z^3-22z^2+2z+33\big)\,y^2+\big(6z^3+26 \\
& z^2+26z+6\big)\,y+11z^2+10z+11\big)\,x^2-2\,\big(z^2\,\big(5z^3+3z^2+3z+5\big)\,y^5+z\,\big(22z^4 \\
& +5z^3-22z^2+5z+22\big)\,y^4+\big(5z^5+5z^4-26z^3-26z^2+5z+5\big)\,y^3+\big(3z^4- \\
& 22z^3-26z^2-22z+3\big)\,y^2+\big(3z^3+5z^2+5z+3\big)\,y+5z^2+22z+5\big)\,x+15z^2+2z \\
& +2y(z-1)^2(z+1)+2y^3(z-1)^2z(z+1)+y^4z^2\,\big(15z^2+2z+15\big)+y^2\,\big(15z^4 \\
& -2z^3-90z^2-2z+15\big)+15\big]\,/\,\big[(x-1)^2(y-1)^2(xy-1)^2(z-1)^2(yz-1)^2 \\
& (xyz-1)^2\big]-\big[4(x+1)(y+1)(yz+1)\,\big(-z^2y^4+4z(z+1)y^3+\big(z^2+1\big)\,y^2 \\
& -4(z+1)y+4x\,\big(y^2-1\big)\,\big(y^2z^2-1\big)+x^2\,\big(z^2y^4-4z(z+1)y^3-\big(z^2+1\big)\,y^2 \\
& +4(z+1)y+1\big)-1\big)\log(x+1)\big]\,/\,\big[(x-1)^3x(y-1)^3(yz-1)^3\big]-\big[4(y+1)(xy \\
& +1)(z+1)\,\big(x^2\,\big(z^2-4z-1\big)\,y^4+4x(x+1)\,\big(z^2-1\big)\,y^3-\big(x^2+1\big)\,\big(z^2-4z-1\big) \\
& y^2-4(x+1)\,\big(z^2-1\big)\,y+z^2-4z-1\big)\log(xy+1)\big]\,/\,\big[x(y-1)^3y(xy-1)^3(z- \\
& 1)^3\big]-\big[4(z+1)(yz+1)\,\big(x^3y^5z^7+x^2y^4(4x(y+1)+5)z^6-xy^3\,\big(\big(y^2+ \\
& 1\big)\,x^2-4(y+1)x-3\big)\,z^5-y^2\,\big(4y(y+1)x^3+5\,\big(y^2+1\big)\,x^2+4(y+1)x+1\big)\,z^4+ \\
& y\,\big(y^2x^3-4y(y+1)x^2-3\,\big(y^2+1\big)\,x-4(y+1)\big)\,z^3+\big(5x^2y^2+y^2+4x(y+1) \\
& y+1\big)\,z^2+((3x+4)y+4)z-1\big)\log(xyz+1)\big]\,/\,\big[xy(z-1)^3z(yz-1)^3(xyz-1)^3\big]\big)\big] \\
& /\,\big[(x+1)^2(y+1)^2(xy+1)^2(z+1)^2(yz+1)^2(xyz+1)^2\big]\;\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z.
\end{aligned}
$$

Table 1. The E_5 integral.

and Borwein, 2011b). We did find a transformation that reduced this to a 3-D integral, but the resulting 3-D integral is extremely complicated (see Table 1). Just converting this expression—originally produced in *Mathematica*—to a working computer program required considerable ingenuity.

The experimental evaluation for E_5 shown in (15) required considerable effort, both computational and analytical. The numerical evaluation of the integral in Table 1 to 240 digits required four hours on 64 CPUs of the Virginia Tech Apple system. Applying PSLQ to the resulting numerical value (together with the numerical values of a set of conjectured terms), yielded (15):

$$E_5 \stackrel{?}{=} 42 - 1984\,\mathrm{Li}_4\left(\frac{1}{2}\right) + \frac{189}{10}\pi^4 - 74\zeta(3) - 1272\zeta(3)\log 2 + 40\pi^2\log^2 2 \\ - \frac{62}{3}\pi^2 + \frac{40}{3}\pi^2\log 2 + 88\log^4 2 + 464\log^2 2 - 40\log 2. \tag{15}$$

This has been confirmed to a full 240 decimal places, namely the precision to which the integral itself was computed, although we do not have a formal proof (nor even a computer-symbolic proof). The numerical computation of a related integral D_5 was even more demanding than E_5. Nonetheless, 18 hours on 256 CPUs of the Apple system at Virginia Tech produced 500 good digits. An even more extensive PSLQ search was performed on the result, but so far we have not been able to find any simplification for this problem. Further attempts to simplify this sum produced no additional results, illustrating that a result derived with PSLQ frequently may have no additional redundancy. ◇

Our tools were initially developed for simplification of constants arising in previous work on *box integrals* performed by Bailey, Borwein and Crandall. The paper (Bailey and Borwein, 2011a) discusses the increasing importance and methodology of such experimental mathematics work. See (Bailey et al., 2010b) and (Borwein et al., 2010) for much more detail on these integrals, their calculation and relevance. A family of integrals crucial to this work is described next:

Example 5 (Integrals III)**.**

$$J(t) := \int_{[0,1]^2} \frac{\log(t + x^2 + y^2)}{(1 + x^2)(1 + y^2)}\,\mathrm{d}x\,\mathrm{d}y. \tag{16}$$

Specification of $t \geq 0$ provides much more strenuous and interesting examples for this kind of simplification. As explained in (Borwein et al., 2010; Borwein and Crandall, Accepted May 2010), for all algebraic t there is in principle a hypergeometric evaluation of $J(t)$. For $t = 0$ one may analytically obtain

$$J(0) = \frac{\pi^2}{16}\log 2 - \frac{7}{8}\zeta(3). \tag{17}$$

For $t = 1$ the initial evaluation for this integral is 210 terms and 12,506 characters in *Mathematica*. Application of the default options of the package reduced this to 58 terms and 3916 characters. ◇

Example 6 (Integrals and polylogarithms)**.** A more challenging constant is $J(3)$, also referred to as $K5$ in the literature (Borwein et al., 2010). A computation in *Mathematica* returned 795 terms, most of which are complex, and 59,040 characters. Our programs reduced this to 127 terms, all of which are real, and 11,539 characters. Consider:

$$
\begin{aligned}
&\frac{\mathrm{Li}_3\left(\frac{1}{2}-\frac{i}{6}\right)}{2}+\frac{\mathrm{Li}_3\left(\frac{1}{2}+\frac{i}{6}\right)}{2}-\mathrm{Li}_3\left(\frac{1}{2}-\frac{i}{2}\right)-\mathrm{Li}_3\left(\frac{1}{2}+\frac{i}{2}\right)-\mathrm{Li}_3\left(\frac{1}{2}-i\right)-\mathrm{Li}_3\left(\frac{1}{2}+i\right)\\
&+\frac{\mathrm{Li}_3\left(\frac{1}{2}-\frac{3i}{2}\right)}{2}+\frac{\mathrm{Li}_3\left(\frac{1}{2}+\frac{3i}{2}\right)}{2}-\mathrm{Li}_3(1-i)-\mathrm{Li}_3(1+i)-\frac{\mathrm{Li}_3\left(\frac{i-\frac{i}{\sqrt{2}}}{i-\frac{1}{\sqrt{2}}}\right)}{2}-\frac{\mathrm{Li}_3\left(\frac{-i+\frac{i}{\sqrt{2}}}{-i-\frac{1}{\sqrt{2}}}\right)}{2}\\
&+\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}-\frac{i}{2}\right)\left(-1+\frac{1}{\sqrt{2}}\right)}{-i-\frac{1}{\sqrt{2}}}\right)}{2}+\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}+\frac{i}{2}\right)\left(-1+\frac{1}{\sqrt{2}}\right)}{i-\frac{1}{\sqrt{2}}}\right)}{2}-\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}-i\right)\left(-1+\frac{1}{\sqrt{2}}\right)}{-2i-\frac{1}{\sqrt{2}}}\right)}{2}\\
&-\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}+i\right)\left(-1+\frac{1}{\sqrt{2}}\right)}{2i-\frac{1}{\sqrt{2}}}\right)}{2}+\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}-\frac{i}{2}\right)\left(-1-\frac{1}{\sqrt{2}}\right)}{-i+\frac{1}{\sqrt{2}}}\right)}{2}+\frac{\mathrm{Li}_3\left(-\frac{i\left(1-\frac{1}{\sqrt{2}}\right)}{-i+\frac{1}{\sqrt{2}}}\right)}{2}-\frac{\mathrm{Li}_3\left(\frac{-i-\frac{i}{\sqrt{2}}}{-i+\frac{1}{\sqrt{2}}}\right)}{2}\\
&+\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}+\frac{i}{2}\right)\left(-1-\frac{1}{\sqrt{2}}\right)}{i+\frac{1}{\sqrt{2}}}\right)}{2}+\frac{\mathrm{Li}_3\left(\frac{i\left(1-\frac{1}{\sqrt{2}}\right)}{i+\frac{1}{\sqrt{2}}}\right)}{2}-\frac{\mathrm{Li}_3\left(\frac{i+\frac{i}{\sqrt{2}}}{i+\frac{1}{\sqrt{2}}}\right)}{2}-\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}-i\right)\left(-1-\frac{1}{\sqrt{2}}\right)}{-2i+\frac{1}{\sqrt{2}}}\right)}{2}\\
&-\frac{\mathrm{Li}_3\left(-\frac{2i\left(1-\frac{1}{\sqrt{2}}\right)}{-2i+\frac{1}{\sqrt{2}}}\right)}{2}-\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}+i\right)\left(-1-\frac{1}{\sqrt{2}}\right)}{2i+\frac{1}{\sqrt{2}}}\right)}{2}-\frac{\mathrm{Li}_3\left(\frac{2i\left(1-\frac{1}{\sqrt{2}}\right)}{2i+\frac{1}{\sqrt{2}}}\right)}{2}+\frac{\mathrm{Li}_3\left(-\frac{i\left(1+\frac{1}{\sqrt{2}}\right)}{-i-\frac{1}{\sqrt{2}}}\right)}{2}\\
&+\frac{\mathrm{Li}_3\left(\frac{i\left(1+\frac{1}{\sqrt{2}}\right)}{i-\frac{1}{\sqrt{2}}}\right)}{2}-\frac{\mathrm{Li}_3\left(-\frac{2i\left(1+\frac{1}{\sqrt{2}}\right)}{-2i-\frac{1}{\sqrt{2}}}\right)}{2}-\frac{\mathrm{Li}_3\left(\frac{2i\left(1+\frac{1}{\sqrt{2}}\right)}{2i-\frac{1}{\sqrt{2}}}\right)}{2}-\mathrm{Li}_3\left(-\frac{2}{-1-\sqrt{2}}\right)\\
&+\frac{\mathrm{Li}_3\left(\frac{-2i-i\sqrt{2}}{-2i+\frac{1}{\sqrt{2}}}\right)}{2}+\frac{\mathrm{Li}_3\left(\frac{2i-i\sqrt{2}}{2i-\frac{1}{\sqrt{2}}}\right)}{2}+\frac{\mathrm{Li}_3\left(\frac{-2i+i\sqrt{2}}{-2i-\frac{1}{\sqrt{2}}}\right)}{2}+\frac{\mathrm{Li}_3\left(\frac{2i+i\sqrt{2}}{2i+\frac{1}{\sqrt{2}}}\right)}{2}\\
&-\frac{\mathrm{Li}_3\left(\frac{2i-2i\sqrt{2}}{2i-\sqrt{2}}\right)}{2}-\frac{\mathrm{Li}_3\left(\frac{-2i+2i\sqrt{2}}{-2i-\sqrt{2}}\right)}{2}-\mathrm{Li}_3\left(-\frac{2}{-1+\sqrt{2}}\right)+\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}-i\right)\left(-1+\sqrt{2}\right)}{-2i-\sqrt{2}}\right)}{2}\\
&+\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}+i\right)\left(-1+\sqrt{2}\right)}{2i-\sqrt{2}}\right)}{2}+\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}-i\right)\left(-1-\sqrt{2}\right)}{-2i+\sqrt{2}}\right)}{2}+\frac{\mathrm{Li}_3\left(-\frac{2i\left(1-\sqrt{2}\right)}{-2i+\sqrt{2}}\right)}{2}-\frac{\mathrm{Li}_3\left(\frac{-2i-2i\sqrt{2}}{-2i+\sqrt{2}}\right)}{2}\\
&+\frac{\mathrm{Li}_3\left(-\frac{\left(\frac{1}{2}+i\right)\left(-1-\sqrt{2}\right)}{2i+\sqrt{2}}\right)}{2}+\frac{\mathrm{Li}_3\left(\frac{2i\left(1-\sqrt{2}\right)}{2i+\sqrt{2}}\right)}{2}-\frac{\mathrm{Li}_3\left(\frac{2i+2i\sqrt{2}}{2i+\sqrt{2}}\right)}{2}+\frac{\mathrm{Li}_3\left(-\frac{2i\left(1+\sqrt{2}\right)}{-2i-\sqrt{2}}\right)}{2}+\frac{\mathrm{Li}_3\left(\frac{2i\left(1+\sqrt{2}\right)}{2i-\sqrt{2}}\right)}{2}.
\end{aligned}
\tag{18}
$$

We had previously manually divided $J(3)$ to segregate the terms with occurrences of the polylogarithm $\mathrm{Li}_n(z)$, which appeared of *order* $n \leq 3$. For instance, in (18) we show only the terms from $J(3)$ involving the *trilogarithm* (Li_3). These terms were extracted using the included function *groupExpressionsByFunctionCategories*. Before simplification, in (18) we have 48 terms, all of which appear complex.

After simplification, the result is a much more manageable 13 real terms:

$$\operatorname{Re}\left(\operatorname{Li}_3\left(\frac{1}{2}-\frac{i}{6}\right)\right)-2\operatorname{Re}\left(\operatorname{Li}_3\left(\frac{1}{2}-\frac{i}{2}\right)\right)-2\operatorname{Re}\left(\operatorname{Li}_3\left(\frac{1}{2}-i\right)\right)+\operatorname{Re}\left(\operatorname{Li}_3\left(\frac{1}{2}-\frac{3i}{2}\right)\right)$$
$$-2\operatorname{Re}(\operatorname{Li}_3(1-i))+\operatorname{Re}\left(\operatorname{Li}_3\left(\frac{1}{12}\left((6-2i)-5\sqrt{2}\right)\right)\right)-\operatorname{Re}\left(\operatorname{Li}_3\left(\frac{1}{18}\left((9+2i)-5\sqrt{2}\right)\right)\right)$$
$$+\operatorname{Re}\left(\operatorname{Li}_3\left(\left(\frac{1}{2}+\frac{i}{6}\right)-\frac{\sqrt{2}}{3}\right)\right)+\operatorname{Re}\left(\operatorname{Li}_3\left(\left(\frac{1}{2}+\frac{i}{6}\right)+\frac{\sqrt{2}}{3}\right)\right)+\operatorname{Re}\left(\operatorname{Li}_3\left(\frac{1}{12}\left((6-2i)+5\sqrt{2}\right)\right)\right)$$
$$-\operatorname{Re}\left(\operatorname{Li}_3\left(\frac{1}{18}\left((9+2i)+5\sqrt{2}\right)\right)\right)-\operatorname{Li}_3\left(-2\left(1+\sqrt{2}\right)\right)-\operatorname{Li}_3\left(-2+2\sqrt{2}\right).$$

Now at the very least, the expression is 'human readable'. We may note that these terms comprise mostly the real parts of complex terms. These can be further simplified manually or using other simplification rules as will be discovered in (Lewin, 1981). As in Example 1 dealing with complex logarithms, care must be taken to take appropriate branches of these functions to get correct results. Thus, the code does not handle such simplifications automatically. The final form found involves terms like those in Example 7, and may be examined in (Borwein et al., 2010). ◇

We observe that simple redundancies or branch issues such as illustrated in Example 3 can and will replicate and grow unmanageably as they have in Example 6.

Example 7 (Complexity reduction)**.** Perhaps the most striking closed form this family of integrals is that of $J(2)$, derived and discussed in (Borwein et al., 2010). This integral starts at about $100,000$ characters and reduces to only a few dozen characters (four simple terms):

$$J(2)=\frac{\pi^2}{8}\ln(2)-\frac{7}{48}\zeta(3)+\frac{11}{24}\pi\operatorname{Cl}_2\left(\frac{\pi}{6}\right)-\frac{29}{24}\pi\operatorname{Cl}_2\left(\frac{5\pi}{6}\right), \tag{19}$$

where Cl_2 is again the *Clausen function* $Cl_2(\theta) := \sum_{n\geq 1}\sin(n\theta)/n^2$ (Cl_2 is the simplest non-elementary Fourier series). As in Example 3 it often arises and can be computed well from $\operatorname{Cl}_2(\theta)=\operatorname{Im}\operatorname{Li}_2(e^{i\theta})$. We challenge the reader to explore the derivation of this formula using the included tools. ◇

5. Future research

Tools such as this may also presage a future when mathematics-rich manuscripts can be automatically (or at least semiautomatically) checked for validity.

For example, we frequently check and correct identities in mathematical manuscripts by computing particular values on the LHS and RHS to high precision and comparing results—and then if necessary use software to repair defects.

As an example, in a study of "character sums" we wished to use the following result derived in (Borwein et al., 2008):

$$\sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\frac{(-1)^{m+n-1}}{(2m-1)(m+n-1)^3} \tag{20}$$
$$\stackrel{?}{=} 4\,\mathrm{Li}_4\left(\frac{1}{2}\right)-\frac{51}{2880}\pi^4-\frac{1}{6}\pi^2\log^2(2)+\frac{1}{6}\log^4(2)+\frac{7}{2}\log(2)\zeta(3).$$

Here $\mathrm{Li}_4(1/2)$ is again a polylogarithmic value. However, a subsequent computation to check results disclosed that whereas the LHS evaluates to $-0.872929289\ldots$, the RHS evaluates to $2.509330815\ldots$. Puzzled, we computed the sum, as well as each of the terms on the RHS (sans their coefficients), to 500-digit precision, then applied the PSLQ algorithm. PSLQ quickly found the following:

$$\sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\frac{(-1)^{m+n-1}}{(2m-1)(m+n-1)^3} \tag{21}$$
$$= 4\,\mathrm{Li}_4\left(\frac{1}{2}\right)-\frac{151}{2880}\pi^4-\frac{1}{6}\pi^2\log^2(2)+\frac{1}{6}\log^4(2)+\frac{7}{2}\log(2)\zeta(3).$$

In other words, in the process of transcribing and 'prettyfying' (20) into the original manuscript, "151" had become "51."

It is quite possible that this error would have gone undetected and uncorrected had we not been able to computationally check and correct such results. While any such error may seem trivial, the reliability and integrity of tables and of resources like the *Digital Library of Mathematica Functions* (Olver et al., 2012) demand such errors be identifiable and correctible. The ability to correct may not always matter, but it can be crucial.

We have largely automated tools to validate and correct expressions, contained in a separate, in progress package titled `VerifyEquality`. We describe our code and underlying heuristic in our final example:

Example 8 (Auto-correction)**.** This example and method arose in checking the paper (Bailey et al., 2010b) for accuracy. One integral explored is

$$\Delta_4(-3)=\int_0^1\cdots\int_0^1\left((r_1-q_1)^2+\cdots+(r_4-q_4)^2\right)^{-3/2}\,dr_1\cdots dr_4\,dq_1\cdots dq_4 \tag{22}$$

This evaluates numerically to ≈ 8.40809. The final closed form of the expression was expressed in the paper as

$$-\frac{128}{15}+\frac{1}{63}\,\pi-8\,\log\left(1+\sqrt{2}\right)-32\,\log\left(1+\sqrt{3}\right)+16\,\log 2+20\,\log 3$$
$$-\frac{8}{5}\,\sqrt{2}+\frac{32}{5}\,\sqrt{3}-32\,\sqrt{2}\arctan\left(\frac{1}{\sqrt{8}}\right)-96\,\mathrm{Ti}_2\left(3-2\,\sqrt{2}\right)+32\,G. \tag{23}$$

Here G is the Catalan number and Ti_2 is a generalized tangent value (another polylog) (Lewin, 1981).

To check the accuracy of this and many like formulas, the TeX sourcecode for the closed form was imported into *Mathematica*. Using the import features is faster and less prone to transcription errors compared to typing the closed form in *Mathematica* format. Then the formula itself was evaluated numerically.

In this case, the expression evaluated to ≈ -8.2970, indicating an error. PSLQ was applied to the terms of the sum, which returned

$$-\frac{128}{15} + \frac{16}{3}\,\pi - 8\,\log\left(1+\sqrt{2}\right) - 32\,\log\left(1+\sqrt{3}\right) + 16\,\log 2 + 20\,\log 3$$
$$-\frac{8}{5}\,\sqrt{2} + \frac{32}{5}\,\sqrt{3} - 32\,\sqrt{2}\arctan\left(\frac{1}{\sqrt{8}}\right) - 96\,\mathrm{Ti}_2\left(3 - 2\,\sqrt{2}\right) + 32\,G. \tag{24}$$

This expression evaluates to the correct numerical value, and so indicated a transcription error in the coefficient of π, which changed from "$\frac{16}{3}$" to "$\frac{1}{63}$". Such errors are common in human transcription and in prettifying of machine-generated expressions, and so we seek to automate this process. ◇

To accomplish this automation, `VerifyEquality` first imports the TeX sourcecode for an equation directly from the manuscript using the built in parser. (The file will need minor manual manipulation to display an equality which can be parsed into sides that can be evaluated numerically.) The values are computed and compared. If they do not numerically agree, then PSLQ is run to try to re-extract the true intended relationship. If this fails, the user is presented with the expression, which can be manually checked for correct parsing. Then PSLQ can be run again if desired.

A preliminary version is working, but it has some limitations. For example, certain functions are not automatically interpreted correctly, especially those that are not part of built-in routines. And differences in typing may cause unexpected parsing errors.

For example, in (23) the term "$16\,\log 2 + 20\,\log 3$" omits parentheses of the argument of the logarithms for readability. But the parser does not take this into account, and instead assumes that l, o and g are variables. Upon import this expression becomes $(16\cdot 2 + 20\cdot 3)\cdot l\cdot o\cdot g$. This error prevents the tool from being able to repair the relation — e.g., by manually changing to $\log(2)$ and $\log(3)$. But users cannot be expected to dig through parsed expressions to notice such errors. Thus, improving such facilities is a necessary goal for publication of this work. A further goal is to be able to automatically extract formulas from a paper, eliminating the need for users to manually annotate TeX source files.

Acknowledgements

The authors should like to thank Richard Crandall for his gracious help with sample constants. All feedback is appreciated. Comments on user experience, results, bug-reports, and results should be sent to `adkaiser@gmail.com`.

References

Bailey, D. H., Borwein, J. M., Nov. 2011a. Exploratory experimentation and computation. Not. Amer. Math. Soc. 58, 1410–1419.

Bailey, D. H., Borwein, J. M., 2011b. High-precision numerical integration: Progress and challenges. J. Symb. Comput. 46, 741–754.

Bailey, D. H., Borwein, J. M., Broadhurst, D., Zudilin, W., 2010a. Experimental mathematics and mathematical physics. Gems in Exp. Math., Contemp. Math. 517, 41–58.

Bailey, D. H., Borwein, J. M., Crandall, R. E., 2010b. Advances in the theory of box integrals. Math. Comput. 79 (271), 1839–1866.
URL http://dx.doi.org/10.1090/S0025-5718-10-02338-0

Bailey, D. H., Broadhurst, D. J., 2000. Parallel integer relation detection: Techniques and applications. Math. Comput. 70, 1719–1736.

Beaumont, J., Bradford, R., Davenport, J. H., 2003. Better simplification of elementary functions through power series. In: Proc. 2003 Int. Symp. on Symb. and Algebr. Comp. ISSAC '03. ACM, New York, NY, USA, pp. 30–36.
URL http://doi.acm.org/10.1145/860854.860867

Beaumont, J. C., Bradford, R. J., Davenport, J. H., Phisanbut, N., 2004. A poly-algorithmic approach to simplifying elementary functions. In: Proc. 2004 Int. Symp. on Symb. and Algebr. Comp. ISSAC '04. ACM, New York, NY, USA, pp. 27–34.
URL http://doi.acm.org/10.1145/1005285.1005292

Borwein, D., Borwein, J. M., Straub, A., Wan, J., In press 2012. Log-sine evaluations of Mahler measures, Part II. Integers.

Borwein, J. M., Chan, O.-Y., Crandall, R. E., 2010. Higher-dimensional box integrals. Exp. Math. 19 (4), 431–446.
URL http://dx.doi.org/10.1080/10586458.2010.10390634

Borwein, J. M., Crandall, R. E., Accepted May 2010. Closed forms: what they are and why we care. Not. Amer. Math. Soc.
URL http://www.carma.newcastle.edu.au/jon/closed-form.pdf

Borwein, J. M., Zucker, I. J., Boersma, J., 2008. The evaluation of character Euler double sums. Ramanujan J. 15, 377–405.

Bradford, R., Davenport, J. H., 2002. Towards better simplification of elementary functions. In: Proc. 2002 Int. Symp. on Symb. and Algebr. Comp. ISSAC '02. ACM, New York, NY, USA, pp. 16–22.
URL http://doi.acm.org/10.1145/780506.780509

Buchberger, B., Loos, R., 1982. Algebraic Simplification. In: Buchberger, B., Collins, G. E., Loos, R. (Eds.), Computer Algebra - Symbolic and Algebraic Computation. Springer Verlag, Vienna - New York, pp. 11–43.

Carette, J., 2004. Understanding expression simplification. In: Proc. 2004 Int. Symp. on Symb. and Algebr. Comp. ISSAC '04. ACM, New York, NY, USA, pp. 72–79.
URL http://doi.acm.org/10.1145/1005285.1005298

Casas, R., Fernández-Camacho, M.-I., Steyaert, J.-M., 1990. Algebraic simplification in computer algebra: an analysis of bottom-up algorithms. Theor. Comput. Sci. 74 (3), 273–298.
URL http://www.sciencedirect.com/science/article/pii/030439759090078V

Caviness, B. F., Apr. 1970. On canonical forms and simplification. J. ACM 17 (2), 385–396.
URL http://doi.acm.org/10.1145/321574.321591

Caviness, B. F., Fateman, R. J., 1976. Simplification of radical expressions. In: Proc. third ACM Symp. on Symb. and Algebr. Comp. SYMSAC '76. ACM, New York, NY, USA, pp. 329–338.
URL http://doi.acm.org/10.1145/800205.806352

Dingle, A., Fateman, R. J., 1994. Branch cuts in computer algebra. In: Proc. 1994 Int. Symp. on Symb. and Algebr. Comp. ISSAC '94. ACM, New York, NY, USA, pp. 250–257.
URL http://doi.acm.org/10.1145/190347.190424

Fateman, R., 2003. High-level proofs of mathematical programs using automatic differentiation, simplification, and some common sense. In: Proc. 2003 Int. Symp. on Symb. and Algebr. Comp. ISSAC '03. ACM, New York, NY, USA, pp. 88–94.
URL http://doi.acm.org/10.1145/860854.860883

Fateman, R. J., 1972. Essays in algebraic simplification. Tech. rep., Cambridge, MA, USA.

Fitch, J. P., 1973. On algebraic simplification. The Comput. J. 16 (1), 23–27.
URL http://comjnl.oxfordjournals.org/content/16/1/23.abstract

Gutierrez, J., Recio, T., 1998. Advances on the simplification of sinecosine equations. J. Symb. Comput. 26 (1), 31–70.
URL http://www.sciencedirect.com/science/article/pii/S0747717198902000

Harrington, S. J., 1979. A new symbolic integration system in reduce. The Comput. J. 22 (2), 127–131.
URL http://comjnl.oxfordjournals.org/content/22/2/127.abstract

Hearn, A. C., 1971. Reduce 2: A system and language for algebraic manipulation. In: Proc. second ACM Symp. on Symb. and Algebr. Manip. SYMSAC '71. ACM, New York, NY, USA, pp. 128–133.
URL http://doi.acm.org/10.1145/800204.806277

Kauers, M., 2006. Sumcracker: A package for manipulating symbolic sums and related objects. J. Symb. Comput. 41 (9), 1039–1057.
URL http://www.sciencedirect.com/science/article/pii/S0747717106000502

Lewin, L., 1981. Polylogarithms and associated functions. North Holland.

Maple, 2012. Maple. Maplesoft, Waterloo ON, Canada.
URL http://www.maplesoft.com/support/help/Maple/view.aspx?path=simplify/details

Maxima, 2011. Maxima, a computer algebra system. version 5.25.1.
URL http://maxima.sourceforge.net

Monagan, M., Pearce, R., 2006. Rational simplification modulo a polynomial ideal. In: Proc. 2006 Int. Symp. on Symb. and Algebr. Comp. ISSAC '06. ACM, New York, NY, USA, pp. 239–245.
URL http://doi.acm.org/10.1145/1145768.1145809

Moses, J., 1971. Algebraic simplification a guide for the perplexed. In: Proc. second ACM Symp. on Symb. and Algebr. Manip. SYMSAC '71. ACM, New York, NY, USA, pp. 282–304.
URL http://doi.acm.org/10.1145/800204.806298

Olver, F. W. J., Lozier, D. W., Boisvert, R. F., Clark, C. W., 2012. NIST Digital Handbook of Mathematical Functions.
URL http://dlmf.nist.gov

Schneider, C., 2008. A refined difference field theory for symbolic summation. J. Symb. Comput. 43 (9), 611–644.
URL http://www.sciencedirect.com/science/article/pii/S0747717108000047

Stein, W., et al., 2012. Sage Mathematics Software (Version 5.2). The Sage Development Team.
URL http://www.sagemath.org

Stoutemyer, D. R., 2011. Ten commandments for good default expression simplification. J. Symb. Comput. 46 (7), 859–887, special Issue in Honour of Keith Geddes on his 60th Birthday.
URL http://www.sciencedirect.com/science/article/pii/S0747717110001471

Zippel, R., 1985. Simplification of expressions involving radicals. J. Symb. Comput. 1 (2), 189–210.
URL http://www.sciencedirect.com/science/article/pii/S0747717185800146

6. Appendix

6.1. Included Files and Building

Download and unzip the included folder, which contains three *Mathematica* notebooks. Open the `SimplifySums.nb` (available from `http://www.carma.newcastle.edu.au/jon/portal.html`) and evaluate all cells in the notebook. Two simple examples are provided in `Example.nb`, and an example of using and writing rules is contained in `RuleList.nb`.

6.2. Basic usage

The most basic usage of the functions is to call the supplied 'wrapper' function with its default parameters unchanged. Name the constant that is to be simplified x. Then call

```
simplifySum[x]
```

This performs the following steps:

(1) Sets working precision to 500 digits.
(2) Removes terms that are equal, opposites or complex conjugates numerically and symbolically. Performs the appropriate algebra symbolically to maintain equality to the original sum.
(3) Converts complex terms that are numerically real to real datatypes.
(4) Checks whether remaining complex terms sum to zero and delete them if so.
(5) Check whether the imaginary part of remaining complex terms sum to zero and make them real if so.
(6) Repeatedly runs PSLQ on adjacent terms of the sum, removing and replacing terms each time a relationship is found. Removes all terms in the sum in the event of zero determination.
(7) Checks accuracy and print a summary between each major step.
(8) Returns the new expression.

6.3. Advanced usage

As shown below, the function `simplifySum` supports a number of optional arguments which can be customized to perform the desired combination of simplifications. The function header is specified as follows (the variables, types and their semantics are displayed in Table 1).

```
simplifySum[ sum_,
                digits_ : 500,
                evalNumerically_ : True,
                evalSymbolically_ : True,
                checkNumericalReals_ : True,
                checkSumOfComplex_ : True,
                runPslq_ : True,
```

Variable	Type	Meaning
sum	Sum	The sum to simplify
digits	Integer	Number of digits of numerical precision. Note that this must be large (300-500+) to run PSLQ successfully.
evalNumerically	Boolean	Perform numerical comparisons for equality, opposites and conjugates.
evalSymbolically	Boolean	Perform symbolic comparisons for equality, opposites and conjugates.
checkNumericalReals	Boolean	Set complex terms with real part numerically zero to real.
checkSumOfComplex	Boolean	Remove complex terms if they sum to zero. Make complex terms real if their imaginary parts sum to zero.
runPslq	Boolean	Run PSLQ to simplify with integer relations.
categoryNames	False	If this variable is False, apply PSLQ in adjacent blocks.
	List of Strings	If this variable is a list of strings, separate the array to categories.
simplifyWithRules	Boolean	Apply the user supplied list of rules.
ruleList	List of rules	List of *Mathematica* rule objects to apply (or False if no rules).

Table 1. Variables, Types and Meanings

```
                categoryNames_ : False,
                simplifyWithRules_ : False,
                ruleList_ : False ]
```

Additionally, there are two global variables which are used. The first is `outputLevel`. If set to 0, then no output besides warnings and errors is printed. If set to 1, then basic summaries of the computation are printed at each major step. If set to 2, then more information about the sub-steps of the computation is printed, in particular, progress of the redundancy checks and results of each application of PSLQ. This is useful to see that the code is still proceeding on in the case of a long computation. The second is `$MaxExtraPrecision`, which is set to the large value of 1000 and should not be altered without reason.

An illustrative code snippet follows:

Example 9 (Syntax)**.** The following code was used to generate example 1.

```
C1 = (1/12)*Pi^2*Log[-1 - I] + (1/12)*Pi^2*Log[-1 + I] +
```

```
  (1/12)*Pi^2*Log[1/3 - I/3] + (1/12)*Pi^2*Log[1/3 + I/3] -
  (1/8)*I*Pi*Log[2/3 - (2*I)/3]^2 + (1/8)*I*Pi*Log[2/3 + (2*I)/3]^2
digits = 350;
evalNumerically = True;
evalSymbolically = False;
checkNumericalReals = True;
checkSumOfComplex =  True;
runPslq = True;
categoryNames = {"PolyLog[2,", "PolyLog[3,"} ;
simplifyWithRules = False;
ruleList = False;

simplifySum[ C1, digits, evalNumerically, evalSymbolically,
             checkNumericalReals, checkSumOfComplex, runPslq, categoryNames,
             simplifyWithRules, ruleList]
```

The variable 'categoryNames' is used to split the terms for application of PSLQ. When used, this variable is a list of strings. Each term in the sum will be converted to *Mathematica* `InputForm`, then checked for substring matches with the terms in the list. Any function that doesn't match any supplied categories will be placed into a default category.

In this example, the categories are the 'dilogarithm' and 'trilogarithm', so those terms will each have their own category, while all other terms such as ordinary logarithms or any other known constants will be left in the default category.

The user should take care to consider name collisions, as a term will be placed only in the first match found or the default. ◇

6.4. Final comments

The user may also wish to call the functions individually. Each function has its usage is documented in its its opening comments. Illustration of how to call individual functions is included with function 'simplifySum'.

Remark 2 (Simplification rules)**.** If it is desired to simplify using or more user-defined rules, a function that applies those rules to each term in the sum is included. Recall that a *Mathematica* rule takes the following form:

```
old_expression :> new_expression /; condition
```

The condition parameter is optional.

One should consult the examples included with our package or *Mathematica*'s own documentation on rules for more detail. As mentioned in section 3, the code here applies rules to each element of the sum individually. If rules that effect more than one term in a sum are desired, then use the built in functions which operate on more levels of the subexpressions.

That said, *Mathematica* does not divulge much in the way of documentation on its source code. A direct request for more detail led to the response below:

"The general idea behind the `Simplify` and `FullSimplify` heuristics is that they apply a sequence of transformations, keeping the version of the expression that has the smallest complexity found so far. This process is repeated at all subexpression levels.

There are a few thousand of transformations used (of course most of the transformations apply to relatively narrow classes of expressions). We do not have documentation describing the transformations or the exact structure of the `Simplify` and `FullSimplify` heuristics."

An implementation in *Maple* or SAGE would thus be more flexible. $\diamond$

7. Vitae

Can be supplied when and if needed.